APPROACHES TO RTI IMPLEMENTATION OF HLA DATA DISTRIBUTION MANAGEMENT SERVICES

Daniel J. Van Hook, Steven J. Rak, James O. Calvin
MIT Lincoln Laboratory
Lexington, MA 02173

Keywords: RTI, HLA, Multicast, Scalability

ABSTRACT

The RTI needs knowledge of each federate's data requirements in order to support HLA requirements for data distribution management services in an efficient and scalable fashion. Data requirements consist of: 1) publications of the classes, attributes, and attribute values to be produced and 2) subscriptions for the classes, attributes, and attribute values to be consumed, i.e., those that are of interest. The RTI supports a *routing space* abstraction by which federates declare their data requirements. This paper describes the RTI routing space abstraction, programming interface, and approaches to data distribution management services implemented in the prototype RTI as part of the HLA definition process. Additionally, a framework for optimizing data flow between federates is described, along with examples and simulation results supporting development of the RTI prototype implementation.

1.0 INTRODUCTION

The Department of Defense (DoD) High Level Architecture (HLA) RunTime Infrastructure (RTI) is a software component that provides services commonly required by simulation systems. These services include management of time, ownership, objects, federations, data declaration, and data distribution. These services are described more fully in [1,2,3]. This paper addresses three areas: 1) the data distribution management services in the HLA, 2) an initial implementation of these services in a prototype RTI developed to support HLA definition, and 3) a framework for optimizing data distribution along with examples and simulation results supporting development of the RTI prototype.

A goal of the HLA is appropriate support for all types of DoD simulations. Because a diverse collection of simulation types must be supported, the RTI data distribution services must be efficient and flexible. Efficiency requires that the RTI mechanisms and interfaces be suited to scaling of simulations from very small to very large along many dimensions including numbers of simulated objects, complexity of interactions, fidelity of representations, and computational/network resources. Flexibility requires that the RTI mechanisms and interfaces not be tied to any particular problem domain or technology but instead be general in nature. Simultaneous requirements for efficiency and flexibility are often difficult to satisfy and solutions that address either or both must also be tempered by economic realities.

Data distribution in the DIS protocols and architecture has been for the most part broadcast, i.e., updates and interactions are routed from each producing simulation to all other simulations. Irrelevant information is generally discarded at the receiving simulations in order to reduce local processing loads. This results in obvious inefficiencies: local processing resources are consumed to filter out irrelevant data; network resources are consumed to deliver data that will only be thrown away. For small scale simulations such waste is relatively unimportant. However, the DIS broadcast scheme becomes unworkable as exercises increase beyond a relatively small size.

These scaling problems have been addressed in various ways as part of a number of recent programs and demonstrations including STOW-E (Synthetic Theater Of War - Europe) [4], STOW RITN (Realtime Information Transfer and Networking) [5,6], and JPSD (Joint Precision Strike Demonstration) [7]. Common characteristics shared to a greater or lesser extent by these approaches include:

- Multicast addressing to route all relevant data and minimal irrelevant data from producers to consumers.
- Expression of data interests.
- Hierarchical filtering.
- Hierarchical architecture.

While each of these programs has been reasonably successful in demonstrating the merit of these capabilities for DIS scalability, their infrastructures are tied to the DIS protocol, models, and algorithms. These infrastructures rely upon both implicit and explicit knowledge of the problem domain they support. However, to achieve the objectives of general applicability and flexibility, the HLA cannot depend on domain specific knowledge. The quintessential problem is how to provide the RTI with sufficient information to do efficient data distribution while maintaining domain independence. The remainder of this paper describes aspects of the routing space approach, which appears to satisfy this requirement.

2.0 ROUTING SPACES

A fundamental abstraction of the HLA data distribution management services is the *routing space*. A routing space is a multidimensional coordinate system in which federates express interest for either receiving attributes and interactions from other federates or sending attributes and interactions to other federations. Federates also agree to maintain associations of data they own to routing spaces. The RTI uses federates' expressions of interest to establish the network connectivity needed to distribute all relevant data and minimal irrelevant data from producers to consumers. "Connectivity" as used in this paper is a generic term that covers various schemes such as IP multicast, multiple TCP connections, ATM virtual circuits, etc.

Subscription regions define a subset of a routing space and specify the data that a federate wishes to receive. *Update regions* also define a subset of a routing space and specify data that a federate offers to produce and send. The RTI determines which federates need to receive data from which other federates by detecting when senders' update regions overlap with receivers' subscription regions in a routing space.

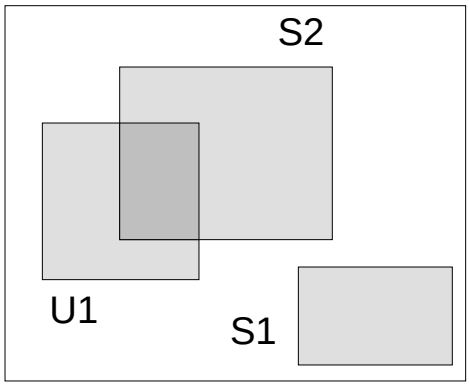

Figure 1: Update (U1) and subscription (S1, S2) regions defined in a two dimensional routing space.

Figure 1 shows a two dimensional routing space with a few regions defined in it. In this example, three federates have specified one region each in the routing space shown. The first federate specifies an update region U1 that defines data the federate wishes to send. The second and third federates specify subscription regions S1 and S2, respectively. These subscription regions define data these two federates wish to receive. The RTI detects that U1 and S2 overlap and establishes the necessary network connectivity so that the data sent by the federate that specified U1 is delivered to the federate that specified S2. In contrast, the RTI detects no overlap between U1 and S1, so the same data does not need to be delivered to the federate that specified S1.

A federation may define multiple routing spaces, each with different characteristics and employed for different purposes. The following information must be agreed on by the federation members to use a routing space:
• The number of dimensions.
• A variable corresponding to each dimension, termed a routing variable.
• The mapping of the routing variables to routing space coordinates.
Of these three pieces of information, the RTI only needs to be made aware of the number of dimensions. The other two – routing variables and mapping – only concern the federate.

The number of dimensions may vary from one up to a maximum value limited only by the RTI implementation and the underlying technology. There is no conceptual limit to the number of dimensions in a routing space. This permits arbitrarily complex filtering to be specified via routing spaces.

Each dimension has a routing variable identified with it by the federation. Routing variables may be attributes such as the scalar temperature. Components of attributes may also be chosen as routing variables. For example, each of the three components of a vector location attribute (e.g., latitude, longitude, and altitude) may be selected as routing variables for the dimensions of a three dimensional routing space. Routing variables may also be functions of an attribute or attributes. For example, speed, calculated as the magnitude of a vector velocity attribute could be selected as a routing variable. Routing variables are not even required to be attributes or be derived from attributes. Any data item that a federation agrees on can be used as a routing variable (e.g., IP subnet address).

In addition to the number of dimensions in a routing space and the routing variables corresponding to each dimension, a federation must agree on the mapping from routing variables expressed in federation units and data types to routing space coordinates. This mapping requires a federation to agree on the minimum and maximum values of each routing variable along with an arbitrary mapping function. The simplest mapping function is of course a linear mapping but the federation may agree to use an arbitrary function of its choice.

3.0 HLA INTERFACE TO DATA DISTRIBUTION MANAGEMENT SERVICES

Data distribution management services are one of the six service groups in the HLA RTI interface [1]. The following paragraphs of this paper list each data distribution management related service (from Chapter 8 of [1]) along with a brief explanation and description. Examining the programming interface is instructive because it brings to the fore many of the issues involved in actually applying the routing space approach and makes the approach more concrete.

```
void CreateUpdateRegion (
  in SpaceHandle,
  in ExtentSet,
  out Region)
void CreateSubscriptionRegion (
  in SpaceHandle,
  in ExtentSet,
  out Region)
void deleteRegion (
  in Region)
```

Subscription and update regions are created and deleted via these services. The first two services take

as arguments a space handle and an extent set while returning a handle to the new region. The space handle identifies a particular routing space, parameters for which are defined in a configuration file in the prototype RTI. The parameters for a routing space include the number of dimensions and any configuration details specific to the implementation of the routing and filtering algorithms employed. The extent set defines the subset of the routing space to create as a region. The extent set consists of an array of endpoints for extents along each dimension of a routing space. It defines a rectangular region in the routing space: a line segment in one dimension, a rectangle in two dimensions, a box in three dimensions, etc. The endpoints are expressed in normalized integer coordinates that range from 0 to 2^{31} - 1 in the prototype RTI implementation. Each extent in an extent set also includes a floating point rate that the RTI can use to extrapolate extents forward in simulated time if desired. The returned region handle is passed in subsequent calls to reference the newly created region.

```
void   subscribeObjectClassAttribute
(
  in ObjectClassHandle,
  in AttributeHandle,
  in Region)
void subscribeInteractionClass (
  in InteractionClassHandle,
  in Region)
```

Attributes of object classes and interaction classes are subscribed for via these services. A federate may subscribe multiple times using the same region for both attributes and for interactions. Notionally, each subscription region can be thought of as a sensor with some area or volume over which it can sense the attributes and classes that are subscribed for. The RTI uses subscription regions provided via these services to deliver and reflect only attributes and interactions associated with update regions that overlap, as described in previous sections. The concept of *association* is described below.

```
void associateUpdateRegion (
  in Region,
  in ObjectId,
  in AttributeHandleSet)
void associateUpdateRegion (
  in Region,
  in InteractionClassHandle)
void disassociateUpdateRegion (
  in Region,
  in ObjectId)
void disassociateUpdateRegion (
  in Region,
  in InteractionClassHandle)
```

These services *associate* or *dissassociate* an update region, specified by a region handle, with either a set of attributes for a particular object instance or with an interaction class. Attribute values or interaction parameters will be delivered to federates that have subscribed with subscription regions that overlap the associated update region in a routing space. Essentially, these services tell the RTI which data is to be sent over which routes as specified by the update and subscription regions. While a region handle is notionally passed each time attributes are updated or interactions are sent, the associate service separates this out to facilitate efficient implementation. The same region may be associated with attributes for multiple object instances and/or interaction classes.

The association formed by these services can be thought of as a contract between the federate and the RTI. The federate agrees to ensure that the characteristics of the object or interaction which map to the dimensions of the update region fall within the extents of that region. If not, the object's attributes either need to be associated with a different region or the extents of the region must be modified to reflect the new object state. For its part, the RTI agrees to deliver associated attributes and interactions to relevant subscribers.

```
void modifyRegion (
  in Region,
  in ExtentSet)
```

This service modifies the extents of an existing update or subscription region identified by the region handle. The new extent set replaces the current extent set. The extent set is as described above. A federate modifies an extent set if the subset of a routing space relevant to the federate changes. For example, consider a ship steaming along on a straight course. The federate simulating the ship periodically invokes this service to modify subscription regions that model the ship's sensors and update regions associated with attributes published for the ship instance.

```
void changeThresholds (
  in Region,
  out ThresholdSet)
```

This service is invoked by the RTI and provided by federates. It is the means by which the RTI informs a federate about how much its subscription or update regions (the subset of a routing space) may change before the RTI must be informed. This service is invoked at least once when each region is created to supply a threshold set. A threshold set is an array of threshold values, one per routing space dimension. The RTI prototype provides these values in normalized coordinates. A federate denormalizes these values into federation types and units and uses them to determine when the federate's interest changes sufficiently to require notifying the RTI through the modify region service. The threshold values reflect any quantization of the routing space by the RTI. In the earlier example of the ship steaming on a straight course, the thresholds provided by the RTI inform the federate how much each update and subscription region may change before the federate must invoke the modify region service.

4.0 PROTOTYPE RTI FRAMEWORK FOR DATA DISTRIBUTION MANAGEMENT SERVICES

Previous sections of this paper described the HLA 's routing space abstraction and interface. This section goes on to describe an underlying framework that supports implementation of these approaches. This framework forms the basis for the prototype RTI data distribution management services but is not implied by the HLA or the Interface Specification. Key concepts in the prototype RTI's framework for data distribution are:

- Separation of routing and forwarding
- Use of agents and hierarchical architecture
- Hierarchical filtering

4.1 Routing vs. Forwarding

The RTI separates routing of data (establishing network connectivity) from forwarding (sending) of data over those routes in order to minimize latencies and maximize throughput. The RTI establishes the necessary network connectivity in advance of when it is actually needed rather than as the data is transferred. Federates express interest to the RTI in terms of classes, attributes, and subscription and update regions in advance of actually sending or receiving the related data. The RTI uses federates' expressions of interest to establish the necessary connectivity.

4.2 Agents to Facilitate Scalability

Agents are software entities that assist simulations in their tasks. Agents may facilitate simulation processing by taking over compute or communication intensive tasks or by supplying information not readily available to a simulation. The concept and uses of agents in supporting a scalable, hierarchical architecture for distributed simulations were described in [8].

The RTI data distribution framework uses subscription agents to manage federates' expressions of interest and to determine the necessary network connectivity. Each federate communicates its data interests, expressed in terms of classes, attributes, and routing space regions, to a subscription agent. Subscription agents cooperate to determine the connectivity needed to forward data from producers to consumers. The RITN program constructed and successfully tested a subscription agent as part of STOW Engineering Demonstration 1A [5,6]. The RITN subscription agent calculated multicast groups based on its client simulations' dynamic subscriptions and publications.

For small simulation systems or simple connectivity algorithms, each federate has its own subscription agent executing locally. Larger simulation systems requiring more interaction or computation may employ a subscription agent for collections of federates, e.g., a subscription agent per local area network.

4.3 Hierarchical Filtering

The RTI data distribution mechanisms support hierarchical filtering. Hierarchical filtering means the ability to filter out or control delivery of data at different points in a simulation system, depending on what resource tradeoffs a federation chooses to make and what infrastructure capabilities are available. Hierarchical filtering notions have been examined in [9].

There are three opportunities for filtering in a simulation system: at sources, in the network via routing, and at receivers. There are different cost tradeoffs involved in each of these. For example, filtering at receivers (after data has been received) requires only a broadcast capability from the network infrastructure and little sophisticated routing capability but lots of computation at receivers as exercise scale grows. On the other hand, source filtering (sending data only if another simulation needs it) requires relatively sophisticated protocols to inform senders about whether any other simulation

has subscribed for particular data. A third alternative, filtering via the network infrastructure, requires a multicasting capability or emulation of multicasting (e.g., via exploders). Such approaches can yield greatly enhanced scalability as demonstrated by the STOW RITN program [5,6].

The RTI supports all aspects of hierarchical filtering. At the source, a federate can be directed by the RTI to not send particular attributes and/or interactions. The approaches described in Section 5 support exploiting capabilities of the network infrastructure to route only relevant data to receivers. Finally, the RTI filters received data based on a hierarchy of tests that include filtering on:

• Destination address.
• Class and attribute names.
• Update region. The RTI can optionally discard received messages based on update region extents included in each message. This capability provide the ability to do exact filtering despite quantization effects of some connectivity algorithms as described in Section 5.

4.4 Data Distribution Framework

The RTI's data distribution framework is depicted in Figure 2. This figure shows five steps involved in data distribution. Each step is described in the following paragraphs.

Express interest. Interest is expressed by each federate to the RTI for the data to be sent and received. Interest is specified as subscription and update regions in routing spaces together with classes and attributes. Federates communicate their interest to subscription agents.

Cluster. Clustering reduces the number of regions that must be manipulated by combining subscription regions together and combining update regions together. Regions in routing spaces may be clustered (coalesced, combined, unified) by a number of different algorithms. This reduces communication and computation costs related to matching. Clustering is performed by subscription agents.

Match. Matching compares update regions with subscription regions to determine overlap in the routing space. Attributes and interactions associated with update regions must be received by federates whose subscription regions overlap with update regions. Matching produces a list of destination federates for each update region. Such a receiver list may be empty in which case the data associated with that update region need not be forwarded or even calculated in the first place. That data is relevant to no other federate.

Establish connectivity. Network connectivity is established based on sets of receiving federates resulting from matching update and subscription regions. An example of such connectivity is a set of multicast groups.

Transfer data. Attributes and interactions are transferred from producers to consumers using the connectivity that has been established. All relevant data, as determined from matching of clustered subscription and update regions, is relayed to appropriate receivers. As little irrelevant data as possible is routed from senders to receivers.

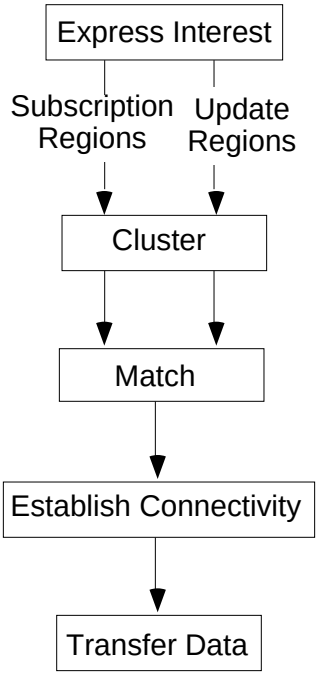

Figure 2: RTI Data Distribution Framework.

4.5 Clustering and Matching By Multidimensional Binary Trees

This section describes an approach to clustering and matching of regions in routing spaces, as outlined in the previous sections. The basic idea of clustering refers to combining (coalescing, combining, unifying) regions in a routing space. The purpose of clustering is to reduce the computation and communication required to match update regions with subscription regions. Clustering takes advantage of the locality of a federate's subscription or update regions in a routing space to produce a more efficient representation.

An important question is how best to represent regions so that they may be efficiently clustered and efficiently matched. A number of approaches applicable to representing spatial regions are described in [10]. A good compromise between factors such as storage, computational cost, complexity, and generality is afforded by the multidimensional equivalent of a binary tree data structure. In two dimensions this is called a quadtree, in three an octtree.

Clustering of regions in routing spaces using multidimensional binary trees is illustrated in Figures 3 and 4. Figure 3 shows a two dimensional routing space with two regions in it denoted R1 and R2. For purposes of this example, the maximum depth of the tree is two so that no further partitioning beyond that shown in the lower right hand quadrant is permitted. Children of nodes are denoted 0, 1, 3, and 2 starting from the upper left quadrant and proceeding in a counter clockwise direction. A leaf node is denoted in this example by referencing the sequence of nodes traversed from the root to the leaf. For example, the bottom right child of the bottom right child of the root node is designated 3–3.

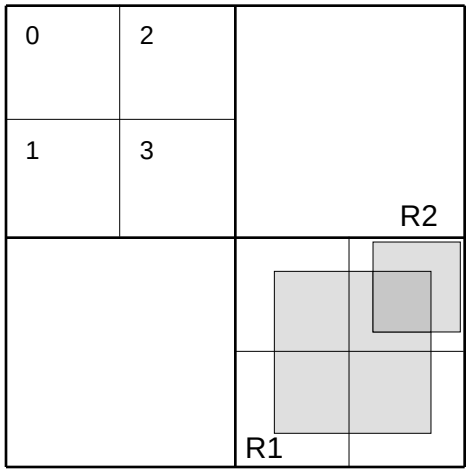

Figure 3: Illustration of clustering of regions in a routing space using multidimension binary trees.

The quadtree data structure corresponding to the routing space in Figure 3 is shown in Figure 4. The ordering of the children at a node is 0, 1, 3, and 2 from left to right. Region R2 is inserted into the quadtree at node 3–2 because R2 lies within the subset of the routing space allocated to this node and only this node. Region R1 is inserted into the quadtree at node 3 because R1 lies within the subsets of all four children of node 3.

Multidimensional binary tree data structures can be very efficiently represented by a set of integer codes that cluster or combine together individual regions inserted in a tree. Such codes reference nodes in the tree and hierarchically define clusters of regions. The code representing the clustering of the regions in our simple quadtree example can be reported as "3." This references the lower right child node of the root. This represents both region R1 and region R2.

Matching consists of efficient mask and compare operations on arrays of integer codes representing subscription regions and update regions clustered via multidimensional binary trees.

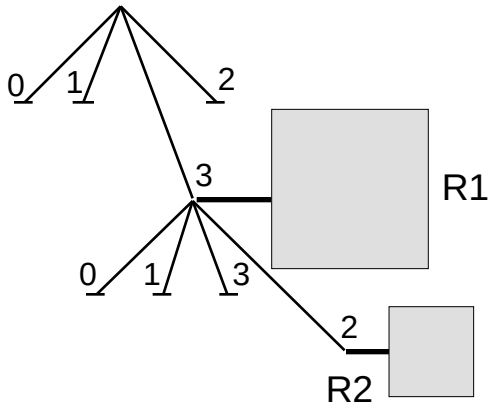

Figure 4: Depiction of the quadtree data structure for the regions and routing space in Figure 3.

5.0 APPROACHES AND ANALYSIS

This section describes some approaches to implementing the data distribution framework described in Section 4. It does not represent testing and evaluation of the performance of the RTI prototype being carried out at IEC/TEC [11]. Instead, it represents results and concepts from design studies carried out to aid in developing the RTI prototype.

The analysis reported in this paper has been performed on log files collected during the STOW RITN ED1A exercise. ED1A consisted of an week long intensive test of the RITN scalability and network infrastructure during which a large body of network traffic was logged. The test scenarios were predominantly ground engagements of between several hundred to more than five thousand simulated entities generated by ModSAF running on 62 workstations. The workstations were distributed across seven LANs at NRL, NRaD, TEC, IDA, and

University of Texas ARL. These sites were connected by a multicasting ATM WAN. Further details about the RITN architecture and the ED1A experiment are found in [5,6]. The data reported in this paper were taken from a scenario consisting of approximately 2000 entities and a little over thirty minutes duration.

various quadtree depths. STOW ED1A Run 2.1 of 14 November 1995.

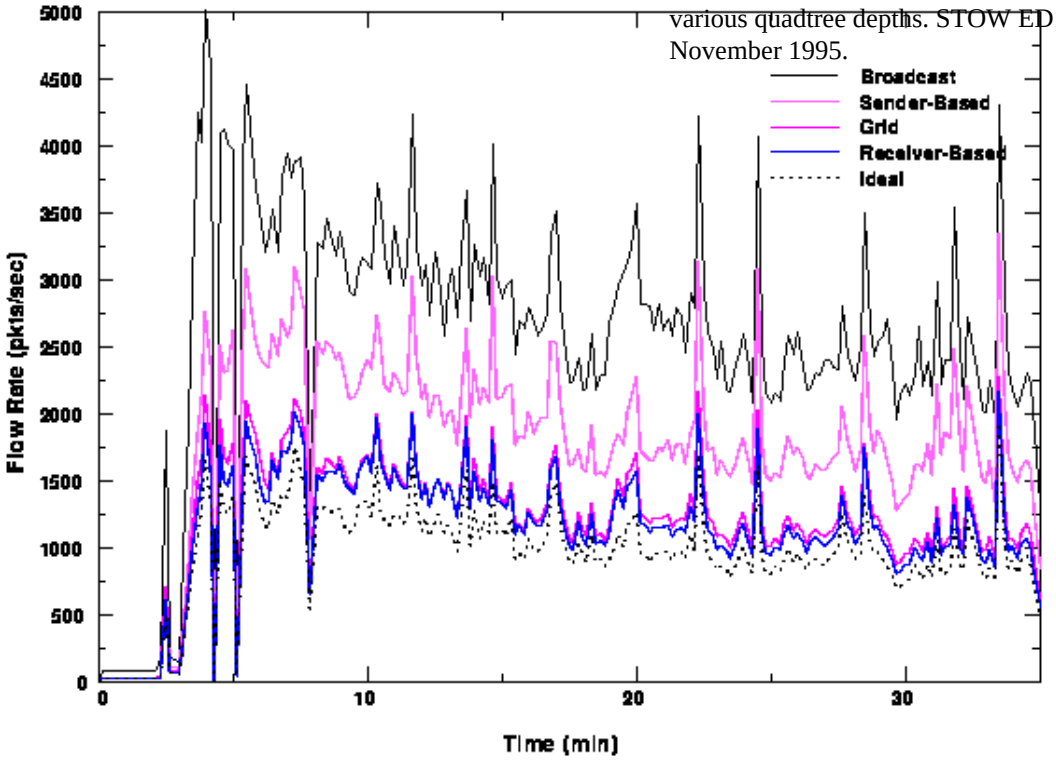

Figure 5: Total packet rates delivered to simulators at the NRaD 156 LAN for various filtering algorithms. STOW ED1A Run 2.1 of 14 November 1995.

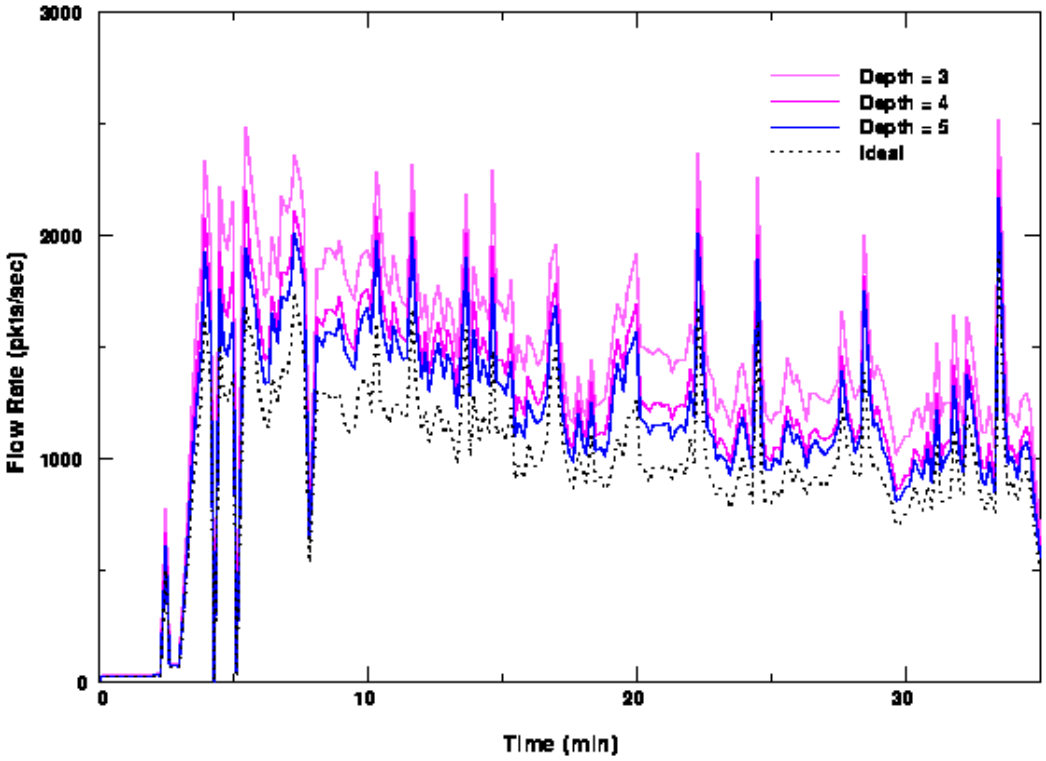

Figure 6: Total packet rates delivered to simulators at the NRaD 156 LAN for receiver-based filtering and

Figure 5 shows the total packet rate delivered to the eleven workstations located at one of the two NRaD LANs over the course of the scenario. The top trace shows the broadcast traffic that would have been delivered to the workstations (the sum over all the workstations) had no filtering algorithms been used. The bottom trace shows ideal traffic flows. Ideal traffic was calculated by post processing logger data to determine what traffic needed to flow between host computers based upon the viewing ranges of the simulated entities. Viewing ranges of four kilometers and ten kilometers for ground and air vehicles were used, respectively.

The broadcast and ideal traces form a useful set of bounds for comparing the effectiveness of other filtering techniques that we wish to evaluate. Three additional classes of techniques (grid-based, receiver-based, and sender-based) will be described and preliminary measures of their performance presented in the following sections. Each of these techniques supports the routing space abstraction, programming interface, and data distribution framework described in previous sections of this paper.

5.1 Grid-Based Filtering

Grid-based filtering approaches provide relatively simple mechanisms for establishing sufficient connectivity for the RTI to route relevant attributes and interactions from producers to consumers. A grid may be used to determine which updates and interactions to route to which receivers as follows:

- Subdivide each routing space into an array of cells and assign a multicast group to each cell.
- Determine which cells overlap each subscription region and join the groups assigned to those cells.
- Determine which cells overlap each update region and send updates for attributes and interactions associated with an update region to the groups assigned to the overlapping cells.
- Federates receive data sent to groups they have joined (relevant data) but do not receive data sent to groups they have not joined (irrelevant data).

Figure 7 illustrates the grid approach. In this example, a two dimensional routing space is configured to have four cells along one dimension and three along the other. A federate specifies a subscription region S1 and joins groups 1–3, 5–7, and 9–11. Attributes and interactions associated with update regions U1 and U2 are sent to groups 10 and

9, respectively and delivered to the federate that specified S1. Attributes and interactions associated with U3 are sent to group 4 but are not delivered to the subscribing federate.

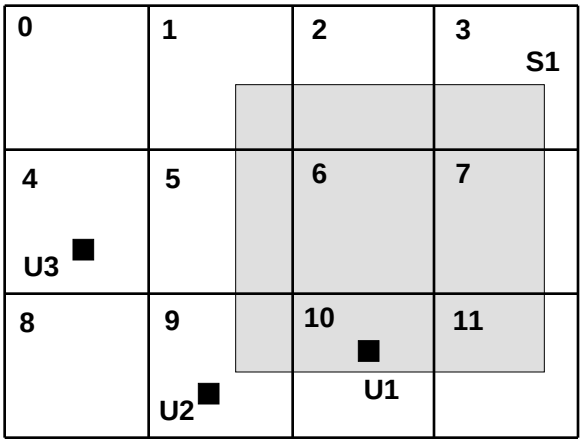

Figure 7: Illustration of grid-based filtering showing pairing of groups with cells and matching of update regions (U1, U2, U3) and subscription regions (S1) via a grid.

Example performance of a grid algorithm with 2.5 kilometer square cells on the STOW ED1A data is shown in Figure 5. As can be seen from the Figure, cumulative delivered traffic is within 10 – 20% of that obtained with the ideal algorithm. The grid departs from ideal performance because of quantization effects of square grids as has been discussed in [12,13].

The simplicity of grid approaches makes them attractive. They are relatively easy to implement and robust because they require no interaction between agents for senders and receivers to establish connectivity. Matching of subscription and update regions is performed implicitly on a grid local to both the senders and receivers. Only the number of cells in each dimension and the algorithm for assigning multicast groups to cells must be shared. Clustering is not absolutely necessary because interest regions need not be communicated nor is mapping of subscription and update regions to grid cells a particularly compute intensive operation.

Despite these good features, grid approaches have a number of drawbacks that make them inadequate or unacceptable for many applications. These include:

• Large numbers of groups may be needed if the number of routing spaces and routing space dimensions is at all large. Multicast groups tend to be a scarce resource whose over use has performance implications in the network infrastructure and in host I/O performance. Use of a larger number of groups also implies an increase in group change rate with more bursty traffic profiles as an undesirable consequence [14].

• Multiple transmissions may be required for extended update regions that are not points in routing space (one for each grid overlapped by the update region) or excessively large interest extents, thereby reducing efficiency. While most vehicles and individual combatants can be adequately dealt with by point update regions, some simulated objects are inherently volumetric. Examples include weather, smoke, dust, electromagnetics, nuclear, chemical, and biological effects.

• No explicit use of information about what data is relevant is exploited, making grid approaches suboptimal. Data is sent even if it is relevant to no other federate. Also, the number of groups used can be in excess of what is actually needed provide the necessary connectivity. This is because senders have no idea what receivers are listening vice versa.

• Irrelevant data is delivered because of the inherent quantization of the grid. Addressing this problem by reducing the grid size (increasing the number of cells along any routing space dimension) can vastly increase multicast group usage. In the RTI prototype, update regions are included in each update to permit irrelevant data to be optionally discarded through filtering at the receivers.

Overall, grid approaches are adequate for applications that do not need to push the limits of scalability, performance, and cost. Relative ease of implementation induced the RTI development team to select a grid approach as the initial approach in the RTI prototype. But better approaches are needed to achieve the required scalability of simulation systems supported by the RTI.

5.2 Receiver-Based Filtering

Receiver-based filtering is distinguished by forming sets of receivers for each piece of data and then establishing the connectivity (e.g. determining a multicast group) to permit that data to be delivered. This is in contrast to the grid-based approach described earlier in which groups were formed with no explicit knowledge of what receivers (if any) were actually interested in a particular piece of data. The motivation for such an approach is that it should yield more effective filtering and control over network and processor resource consumption.

In receiver-based filtering, subscription agents cluster each federates' subscription regions and update

regions and communicate the clustered subscription regions to other subscription agents. Each agent matches its local update regions with subscription regions from other agents to determine a set of receiver federates. The matching operation produces lists of federates to which data associated with each update region must be delivered. Multicast groups providing the necessary connectivity are determined for each update region. Attributes and interactions are transmitted using these groups. As the update regions or subscription regions change the connectivity is incrementally recalculated. A number of approaches are possible for determining what multicast groups to use for each update region. These approaches range from fully static at one extreme to fully dynamic at the other with variations and compromises in between.

A static group database can be established at initialization time and shared by all federates. The group for each update region is determined by looking up an appropriate group, given a list of federates. It is not possible to predefine every possible group for federations composed of more than a small number of workstations. However, tools such as SAT/IAT [15] or Sim^2 Toolset [16] can be used to determine a good set of groups based on expected traffic flows for a particular scenario. At run time, group selection becomes an optimization problem of selecting a group that contains all necessary federates and as few unnecessary federates as possible. In the worst case, a group consisting of all federates is used. This approach is simple and relatively robust but will certainly be less than optimal.

At the other extreme, completely dynamic grouping can be employed. A dynamic scheme creates a new group consisting of precisely the required set of federates. The new group is added to a shared group mapping database. If the necessary group already exists in the shared group mapping database it is not created. This approach can provide near optimal connectivity but may be impractical to implement in a large distributed system unless sophisticated heuristics are employed.

An intermediate approach is to permit the connectivity represented in the shared group mapping database to evolve slowly to match the requirements of each exercise. Such an approach can come close to providing the ideal connectivity while remaining practical.

Initial simulation results for such approaches used a cellified clustering technique instead of multidimensional binary trees and were presented in [17]. Current results for such approaches are shown in Figures 5 and 6. Figure 6 shows the performance of receiver-based filtering for a number of different tree depths relative to ideal filtering. As expected, the effectiveness of the algorithm increases (less traffic is delivered) as the partition size gets smaller (tree depth gets larger). This occurs because less irrelevant traffic is being forwarded to each group as the partition size is reduced. This simulation was done with rectangular extents in a two dimensional routing space. If regions are specified using shapes that more closely match the geometry of simulated objects' interest (e.g., circles), the effectiveness can be made very close to the ideal by increasing tree depth. This is possible without the explosion in multicast groups that would be experienced with a grid-based approach. Figure 5 shows performance of the receiver-based prototype with other approaches.

5.3 Sender-Based Filtering

Sender-based filtering can be considered a dual of receiver-based filtering. Rather than compose sets of receivers, subscription agents for senders instead assign groups to each update region cluster and inform other agents. Subscription agents acting on behalf of potential receivers match subscription regions with update regions supplied by remote agents to determine which groups to join. The concept is like tuning a radio: agents sample a bit of the signal (the update region) and tune in (join a group) only if they find it interesting.

Group selection can be dynamic or static and is done by the senders' agents. The simplest approach is to statically allocate a set of groups to each subscription agent. The agent clusters update regions and assigns groups to each cluster. Federates send their data to the group for each cluster. Receivers join groups for update region clusters that match their own subscription regions.

Figure 5 shows simulation results for the sender-based filtering approach. This simulation employed a single cluster and group per federate. As the graph shows, this simple approach yields a considerable benefit over broadcast but is not nearly as efficient as grid or receiver-based approaches. However, only a very small number of groups (one per federate, i.e., 62) were required to achieve this result. Enhancements to this simple algorithm should result in considerably better performance.

6.0 CONCLUSIONS

Routing spaces are useful abstraction for simulation programmers to use to specify the data requirements of their simulations. They provide relatively refined control over data distribution while maintaining a useful degree of abstraction. In addition, they provide sufficient information to enable implementation of sophisticated and efficient filtering and routing algorithms in the RTI while maintaining federation independence and flexibility.

The RTI can concurrently support a number of filtering/routing schemes so as to satisfy the differing requirements of various federations and types of data. The new techniques being prototyped and evaluated – receiver-based and sender-based filtering – appear to be very promising and are able to address many of the shortcomings of more familiar grid-based schemes.

7.0 REFERENCES

[1] The RTI interface is defined by the document "Department of Defense High Level Architecture For Simulations Version 1.0 Interface Specification." This document can be found on the World Wide Web at http://www.dmso.mil/projects/hla/.

[2] Calvin, James O., Richard Weatherly, "An Introduction to the High Level Architecture (HLA) Run-Time Infrastructure (RTI)," 96-14-103, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.

[3] McGarry, Stephen M., Paul N. DiCaprio, Richard Weatherly, Annette Wilson, "Design Issues for the High Level Architecture (HLA) Run-Time Infrastructure (RTI) Prototype Version 0.2, " 96-14-104, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.

[4] Van Hook, Daniel J., Michael Newton, David Fusco, James O. Calvin, "An Approach to DIS Scaleability," 94-11-141, Eleventh Workshop on Standards for the Interoperability of Distributed Simulations, September 26-30, 1994.

[5] Calvin, James O., Joshua Seeger, Gregory D. Troxel, Daniel J. Van Hook, "STOW Realtime Information Transfer and Networking System Architecture," 95-12-061, Twelfth Workshop on Standards for the Interoperability of Distributed Simulations, March 13-17, 1995.

[6] Van Hook, Daniel J., David P. Cebula, Steven J. Rak, Carol J. Chiang, Paul N. DiCaprio, James O. Calvin, "Performance of STOW RITN Application Control Techniques," 96-14-157, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.

[7] Powell, Edward T., Larry Mellon, James F. Watson, Glenn H. Tarbox, "Joint Precision Strike Demonstration (JPSD) Simulation Architecture," 96-14-116, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.

[8] Calvin, James O., Daniel J. Van Hook, "AGENTS: An Architectural Construct to Support Distributed Simulation," 94-11-142, Eleventh Workshop on Standards for the Interoperability of Distributed Simulations, Sept 26-30, 1994.

[9] Mellon, Larry, "Hierarchical Filtering in the STOW System," 96-14-087, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.

[10] Foley, Van Dam, Feiner, and Hughes, "Computer Graphics: Principles and Practice," Addison-Wesley, 1990.

[11] Olszewski, Jeff, Larry Mellon, Richard Briggs, "HLA Testbed Declaration Management Experiments," 96-15-096, Fifteenth Workshop on Standards for the Interoperability of Distributed Simulations, September, 1996.

[12] Van Hook, Daniel J., Steven J. Rak, James O. Calvin, "Approaches to Relevance Filtering." 94-11-144,Eleventh Workshop on Standards for the Interoperability of Distributed Simulations, September 26-30, 1994.

[13] Rak, Steven J., Daniel J. Van Hook, "Evaluation of Grid Based Relevance Filtering for Multicast Group Assignment," 96-14-106, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.

[14] Van Hook, Daniel J., James O. Calvin, Joshua E. Smith, "Data Consistency Mechanisms to Support Distributed Simulation," 95-12-059, Twelfth Workshop on Standards for the Interoperability of Distributed Simulations, March 13-17, 1995.

[15] Juliano, Michael, Robert D'Urso, Ben Wise, Edward Powell, "Scenario and Infrastructure analysis to Measure Large-Scale CGF Exercise Performance," 1996 CGF

[16] Van Hook, Daniel J., Deborah J. Wilbert, Richard L. Schaffer, Walter Milliken, Dennis K. McBride, "Scaleability Tools, Techniques, and the DIS Architecture," Proceedings of the 15th Interservice/Industry Training Systems and Education Conference, 1993.

[17] Van Hook, Daniel J., Presentation at the STOW RITN Design Review, December 12, 1994.